SC42030 Control for high resolution imaging
Homework Assignment 3 : Shack Hartmann Sensor

Ali Nawaz

August 27, 2018

1 Shack-Hartmann sensor 1

A Shack Hartmann sensor uses small lenses to measure the local gradient of the wavefront. The gradient is measured through the displacement of the centroid in the focal plane of the lens. The relation between gradient of the wavefront and the displacement of the centroid can be expressed with the aid of Equation 1[1][pg.60]. $\phi(x_i, y_j)$ represents the wavefront at spatial position x_i, y_j. $\eta_x(i,j)$, $\eta_y(i,j)$ represent the measurement noise and the effect from higher order aberrations that differ from just the tilt of the wavefront as seen by aperture at location i,j.

$$\begin{aligned} \Delta x(i,j) &= \kappa_x \frac{\delta\phi(x_i, y_j)}{\delta x} + \eta_x(i,j) \\ \Delta y(i,j) &= \kappa_y \frac{\delta\phi(x_i, y_j)}{\delta y} + \eta_y(i,j) \end{aligned} \tag{1}$$

κ is a geometrical constant concerning the imaging system; its magnitude is determined by the optical parameters such as pupil size, distance between pupil and lens, focal length, diameter of lenslet, wavelength of light under observation etc.

2 Shack-Hartmann sensor 2

A simple Shack-Hartmann sensor with three lenslets is shown with the aid of Figure 1[1]. The objective is to determine the number of modes that the system can detect.

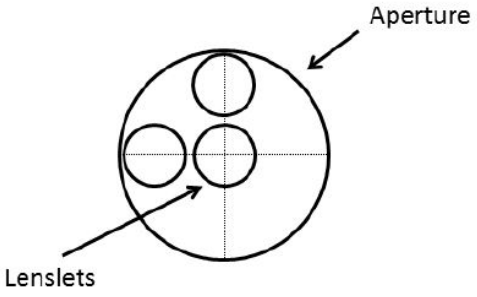

Figure 1: Shack Hartmann sensor with 3 sub apertures.

The number of modes a system can detect, relies on a multitude of factors. This includes the geometry of the aperture, configuration of the lenslets. Focal length, location of CCD sensors, number of cam-

era pixels and other underlying optical performance factors. According to literature [1], the number of Zernike modes that can be reconstructed with a Shack Hartman Wavefront sensor with N sub-apertures, is N. That is the first three modes of the Zernike polynomials can be detected. In general the number of modes that can be reconstructed is limited by the rank of G, the filtered basis gradient matrix. It contains information concerning the "compatibility" of the mask geometry and basis functions. However an analytic dependence of rank G on the mask geometry is difficult to obtain. A frequency domain analysis is rather simpler. However, for this assignment a relatively simplified approach outlined by the computational task from Section 4.2 is presented. The first three modes of Zernike Polynomial are expressed with the aid of Figures 2-11. It is assumed that the microlenses have a focal length of 0.1 to help visualise the effect of wavefront distortion. The CCD camera is assumed to have infinite resolution. For Z_0^0 (piston), the points are projected at the centroid of the lenslets.

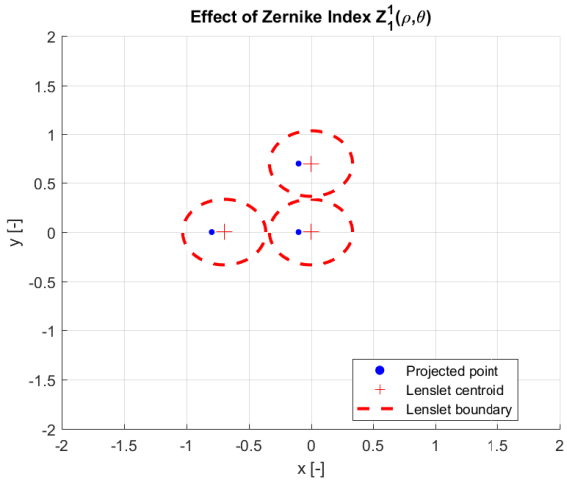

Figure 2: Effect of the Z_1^1(tip) projected image on CCD sensor.

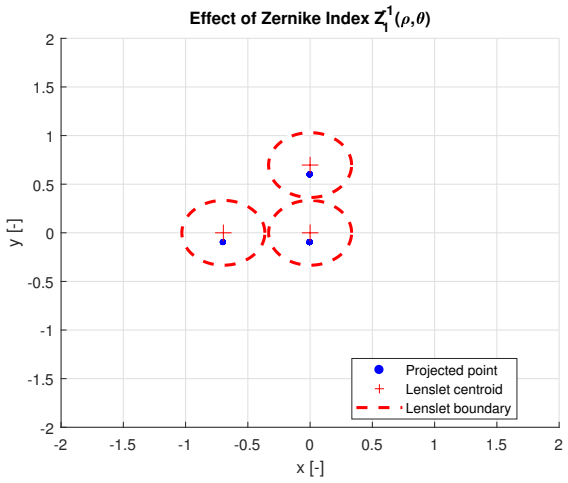

Figure 3: Effect of the Z_1^{-1}(tilt) projected image on CCD sensor.

Beyond these first three modes, the points are no longer concentrated to a point on the CCD camera, but rather scattered. This is presented with the aid of projections till Zernike mode Z_3^{-3}. For some N= 3 and esp. 4, the scattering proceeds into the regions of other lenslet.

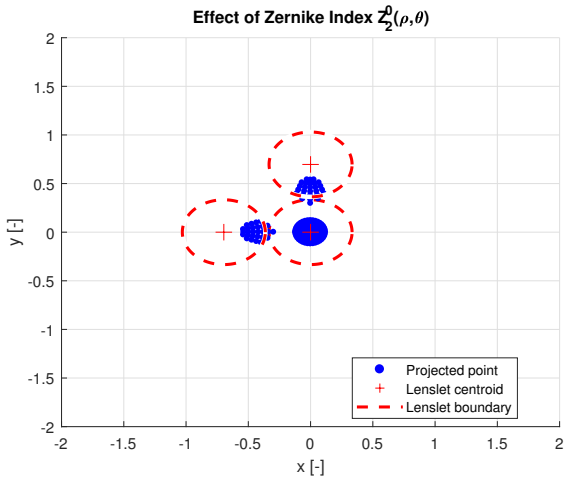

Figure 4: Effect of the Z_2^0(defocus) projected image on CCD sensor.

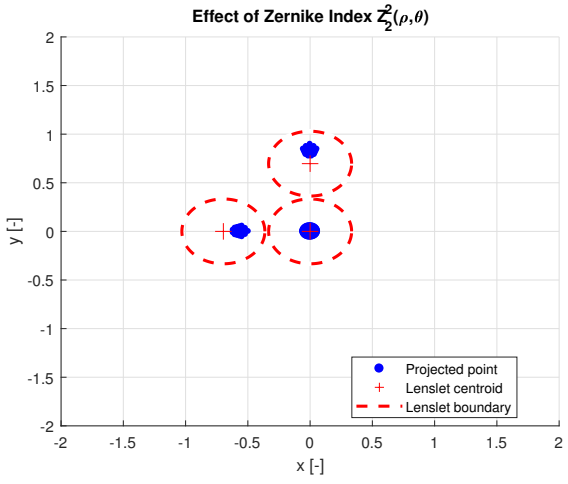

Figure 5: Effect of the Z_2^2(astigmatism) projected image on CCD sensor.

[1] Cerro Tololo Inter-American Observatory: http://www.ctio.noao.edu/ãtokovin/tutorial/part3/wfs.html

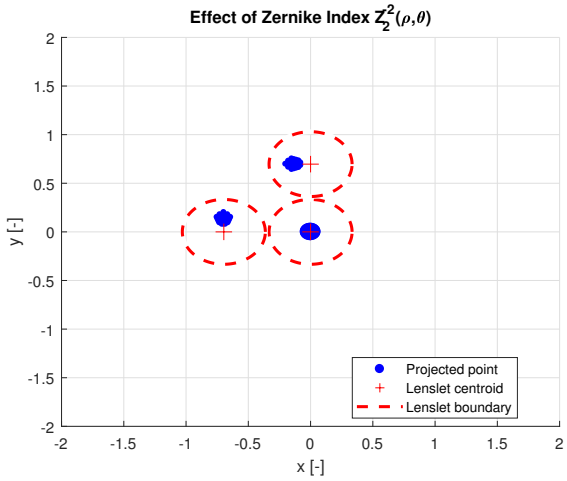

Figure 6: Effect of the Z_2^{-2}(astigmatism) projected image on CCD sensor.

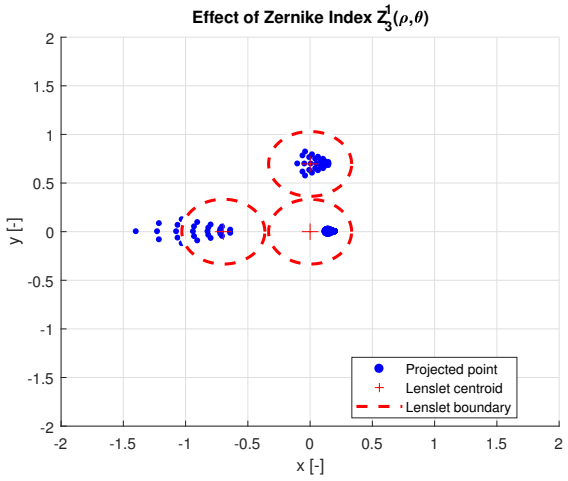

Figure 7: Effect of the Z_3^1(coma) projected image on CCD sensor.

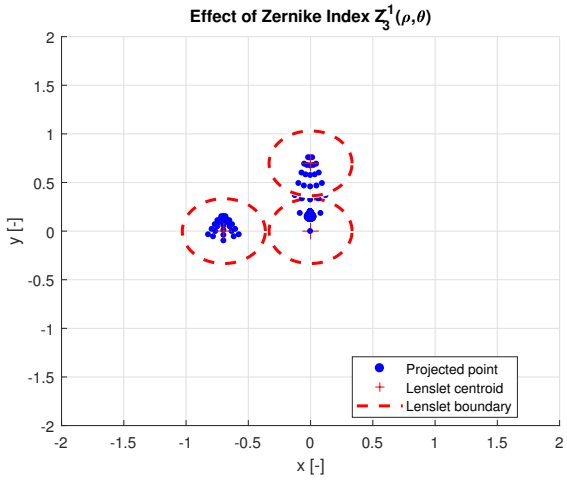

Figure 8: Effect of the Z_3^{-1}(coma) projected image on CCD sensor.

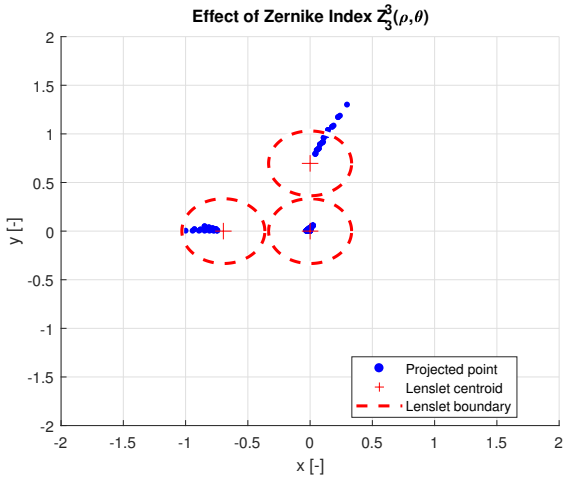

Figure 9: Effect of the Z_3^3(trefoil) projected image on CCD sensor.

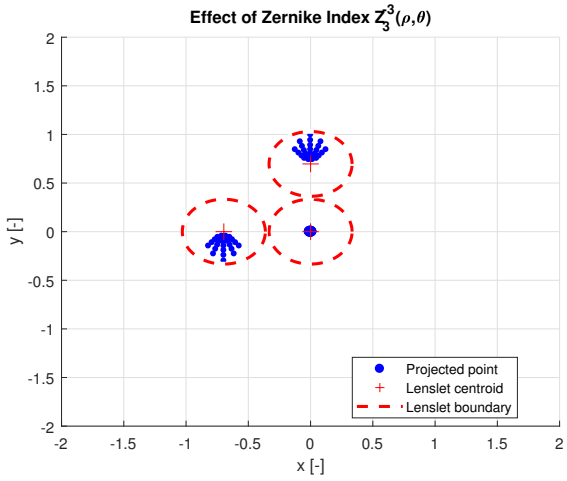

Figure 10: Effect of the Z_3^{-3}(trefoil) projected image on CCD sensor.

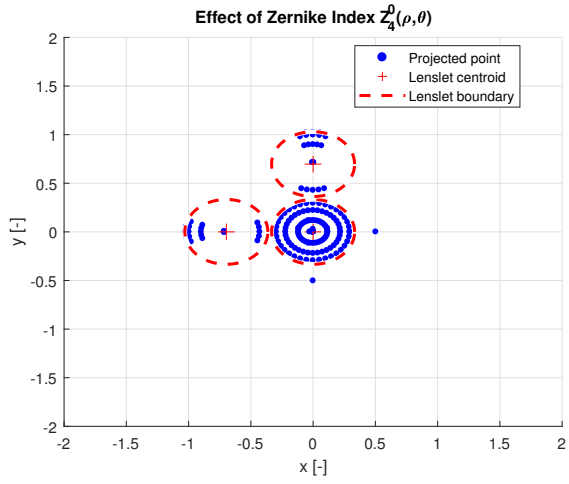

Figure 11: Effect of the Z_4^0(spherical) projected image on CCD sensor.

It is important to note that, the patterns formed rely on multitude of factors, esp. the focal distance. As the focal distance gets closer and closer together, the image is seen as a point, within the boundary of lenslets yet scattered for higher orders.

3 Shack Hartmann sensor 3

Figure 12 outlines a Shack Hartmann sensor with 7 sub apertures[1] arranged in a hexagonal formation. The objective is to find the number N of detectable modes that the sensor can detect. This is again done with the aid of algorithms outlined in Section 4.2. Similar to previous section, a focal length of 0.1 is selected. Radius of lenslet is taken at 1/3, minimum spacing between lenslets is taken to be 0.03 and the CCD sensor is assumed to have infinite resolution for image detection. It is further assumed that the CCD sensor is placed on the focal plane. This presents a very quick method of analysing the project behaviour on the image plane. For proper analysis, the rank of G matrix as discussed in the previous section must be checked for.

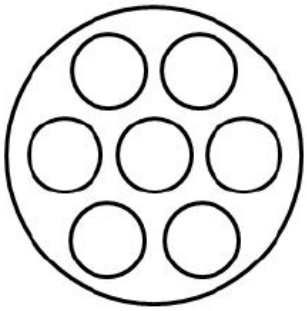

Figure 12: Shack Hartmann sensor with 7 sub apertures.

Figures 13 to 27 represent the behaviour for Zernike polynomials for N = 0 to 4. Apart from piston tip and tilt modes are detected well. Beyond that image distortion divergence into neighbouring lenslet zone is observed. For astigmatism (N = 2, m = 2, -2) and trefoil (N= 3, m= -3), though the projected image points lie inside the lenslet, they are still scattered. Excluding piston for 5 cases the projected image is

observed to be within the region defined by the lenslets, these include tip, tilt, astigmatism (N=2, m=-2,2) and trefoil. If the focal point is brought closer, then 2 more cases Z_4^{-4} and Z_4^4 is observed within the bounds of lenslet. This is presented with the aid of Figure 28, where the focal length is reduced to 0.05 from 0.1. With this it can be concluded that it is not only the area of sub apertures and the configuration of lenslet arrangement that effect the image formed on the observation plane. It further relies on other properties e.g. focal length of lenslet, placement of CCD sensor and the resolution of camera etc.

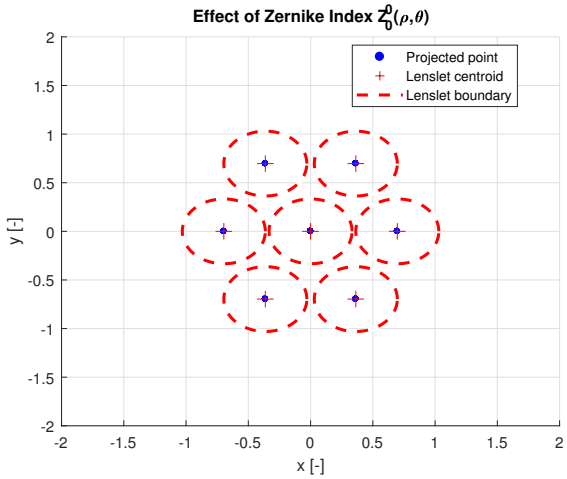

Figure 13: Effect of the Z_0^0(piston) projected image on CCD sensor.

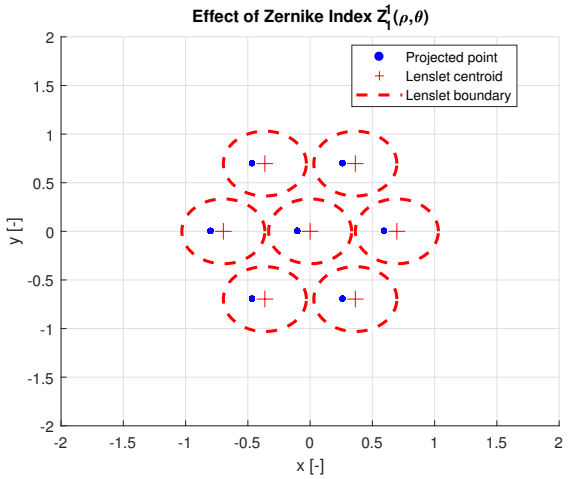

Figure 14: Effect of the Z_1^1(tip) projected image on CCD sensor.

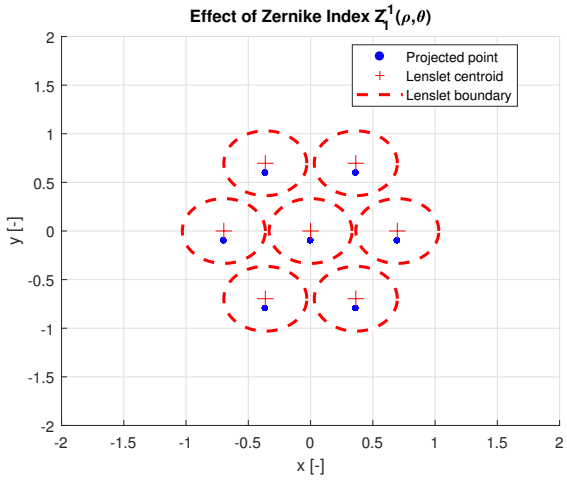

Figure 15: Effect of the Z_1^{-1}(tilt) projected image on CCD sensor.

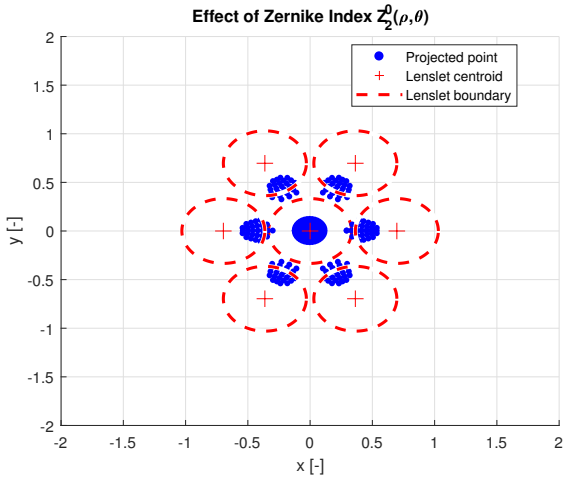

Figure 16: Effect of the Z_2^0(defocus) projected image on CCD sensor.

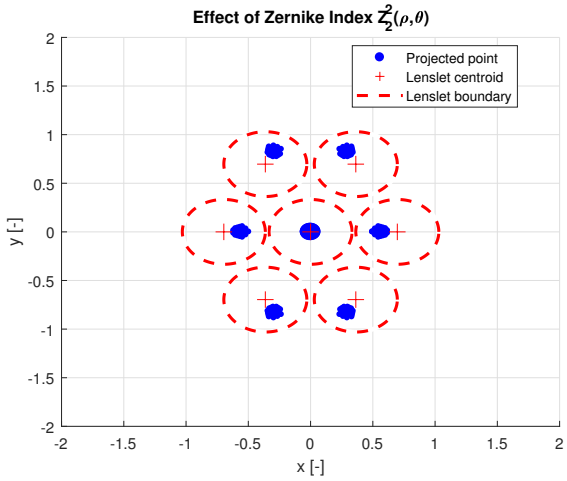

Figure 17: Effect of the Z_2^2(astigmatism) projected image on CCD sensor.

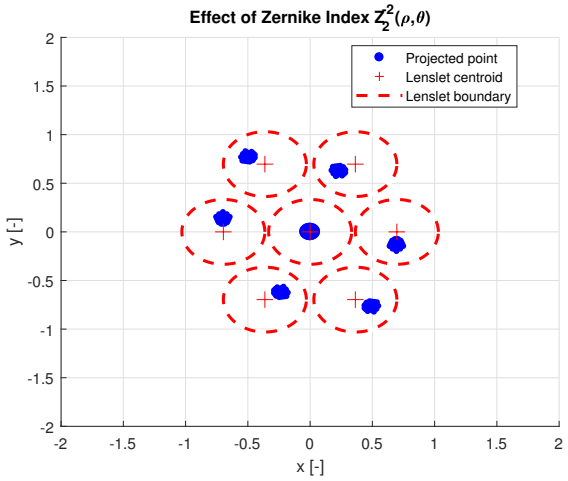

Figure 18: Effect of the Z_2^{-2}(astigmatism) projected image on CCD sensor.

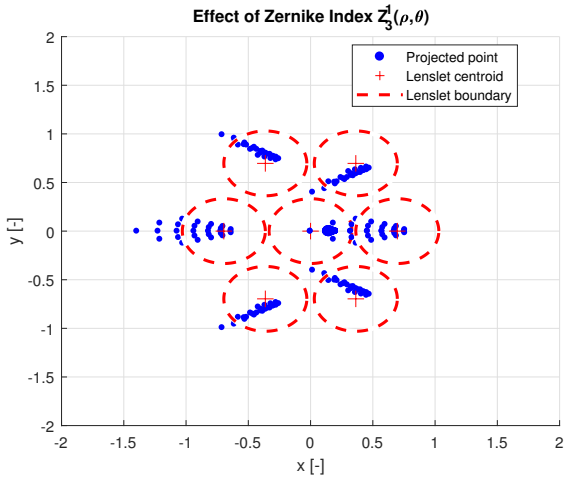

Figure 19: Effect of the Z_3^1(coma) projected image on CCD sensor.

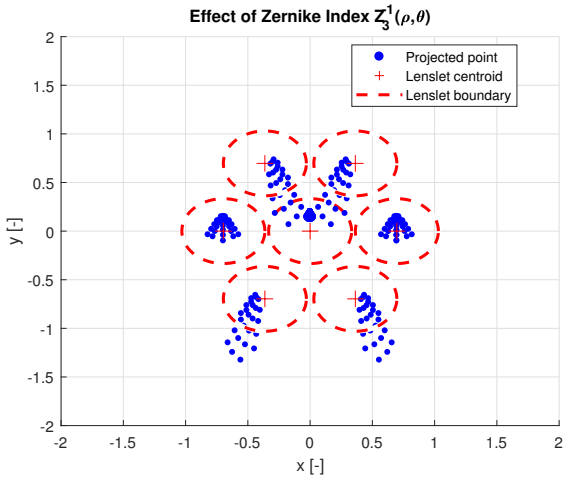

Figure 20: Effect of the Z_3^{-1}(coma) projected image on CCD sensor.

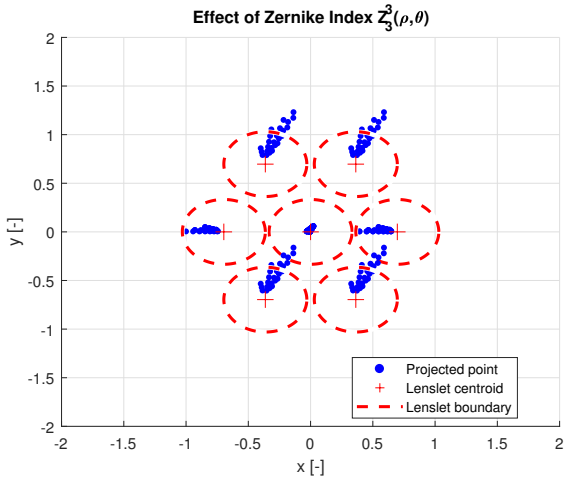

Figure 21: Effect of the Z_3^3(trefoil) projected image on CCD sensor.

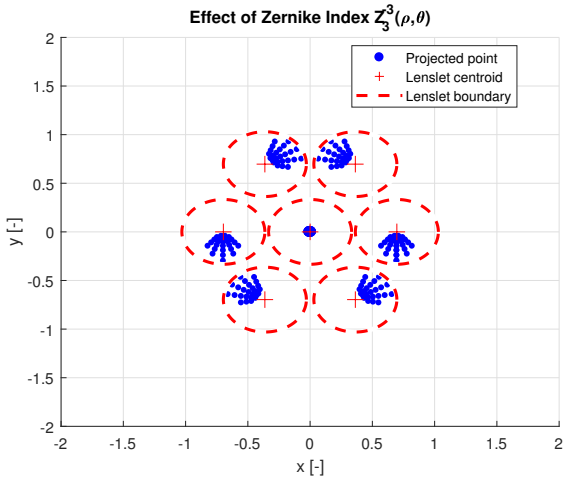

Figure 22: Effect of the Z_3^{-3}(trefoil) projected image on CCD sensor.

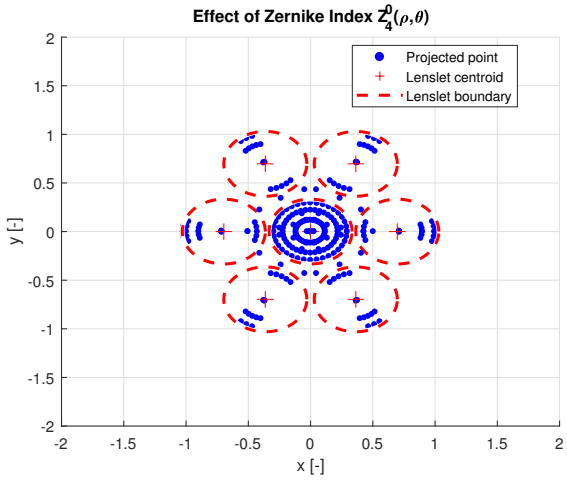

Figure 23: Effect of the Z_4^0(spherical) projected image on CCD sensor.

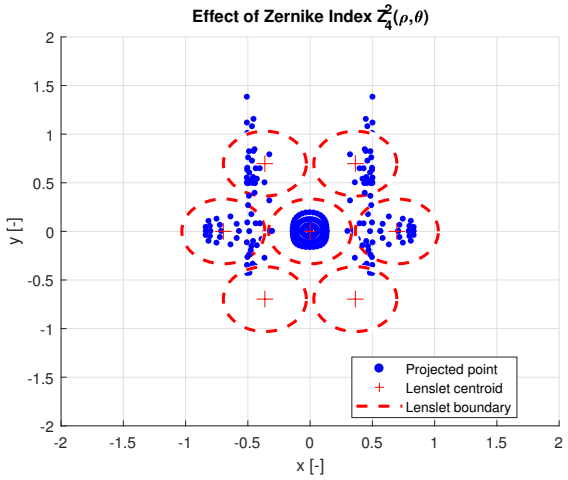

Figure 24: Effect of the Z_4^2 projected image on CCD sensor.

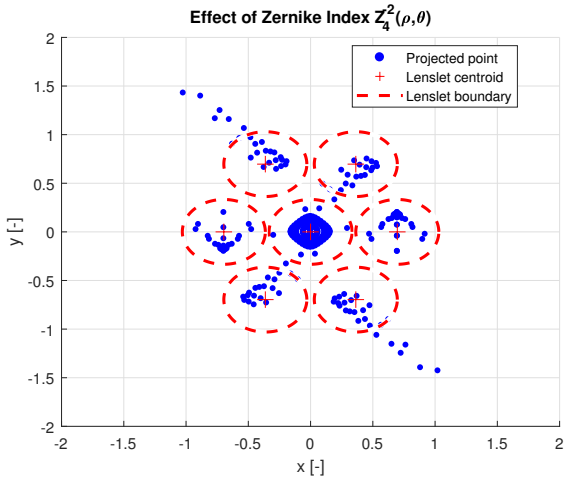

Figure 25: Effect of the Z_4^{-2} projected image on CCD sensor.

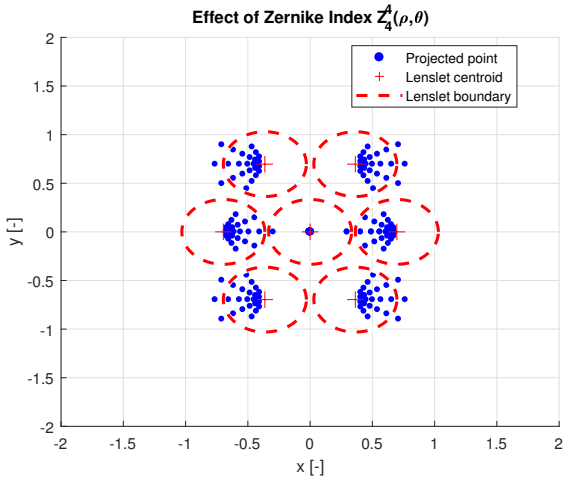

Figure 26: Effect of the Z_4^{4} projected image on CCD sensor.

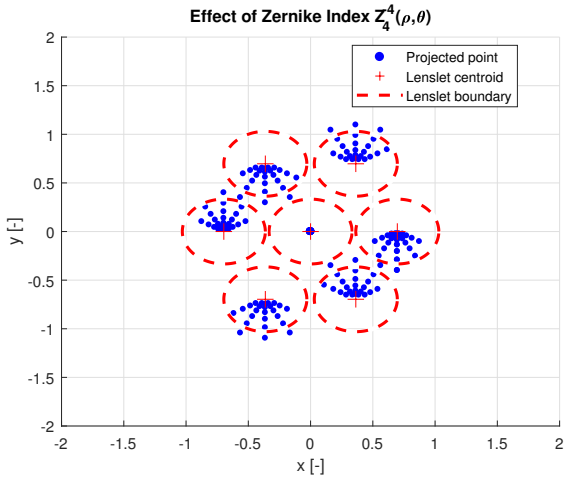

Figure 27: Effect of the Z_4^{-4} projected image on CCD sensor.

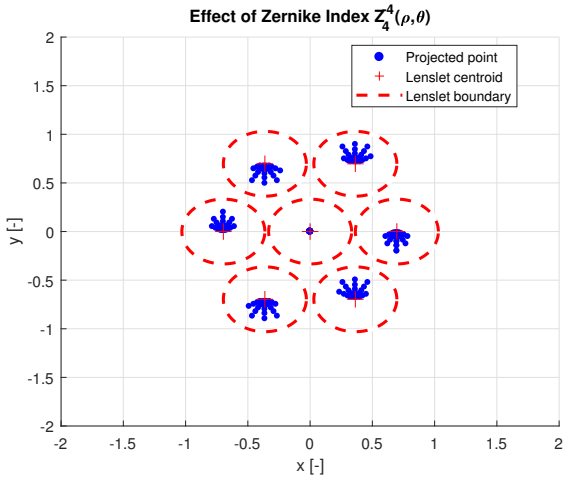

Figure 28: Effect of the Z_4^{-4} projected image on CCD sensor for f= 0.05.

4 Computational Task

This section aims at presenting the algorithms used throughout this assignment. Section 4.1 presents the generation of Zernike modes. Section 4.2 outlines the working principles behind simulating Zernike in circular apertures.

4.1 Generating Zernike modes

The objective is to construct a script capable of computing the value Zernike polynomial for given index N, radial position ρ and angular position θ. Zernike polynomials are defined using polar co-ordinates ρ and θ. For any two non negative integers of the same parity m,n[1][pg.22]:

$$m, n \in Z_{\geqslant 0},\ n \geqslant m,\ n - m \in 2Z \tag{2}$$

The Zernike polynomials are expressed as the product of radial Zernike polynomial $R_n^m(\rho)$ and azimuthal harmonic component of frequency m. This is outlined in Equation 3. A Matlab script used for simulating the Zernike polynomials is presented in Section 5.

$$
\begin{aligned}
&\text{For m} = 0 \\
&Z_n^0 = R_n^m \\
&\text{For } m > 0 \\
&Z_n^m = R_n^m \cdot \cos(m\theta) \\
&\text{For } m < 0 \\
&Z_n^m = R_n^m \cdot \sin(m\theta) \\
&\text{Where the radial polynomial is given by:} \\
&R_n^m(\rho) = \sum_{s=0}^{\frac{n-m}{2}} \frac{(-1)^s (n-s)!}{s!(\frac{n+m}{2}-s)!(\frac{n-m}{2}-s)!} \rho^{n-2s}
\end{aligned}
\tag{3}
$$

The script is used to generate images of Zernike polynomials in the first 4 orders. This is presented in Figures 29 - 43.

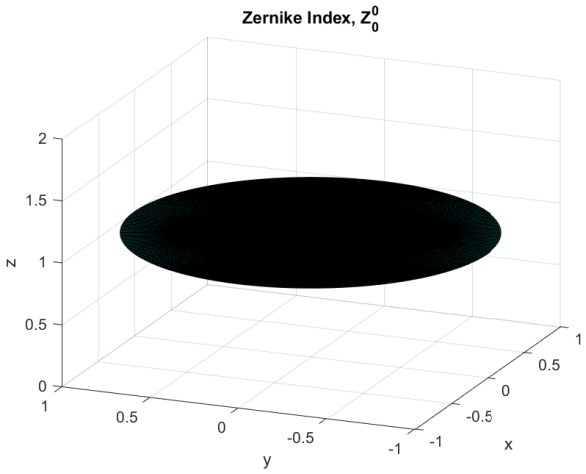

Figure 29: Wavefront for Z_0^0(piston) .

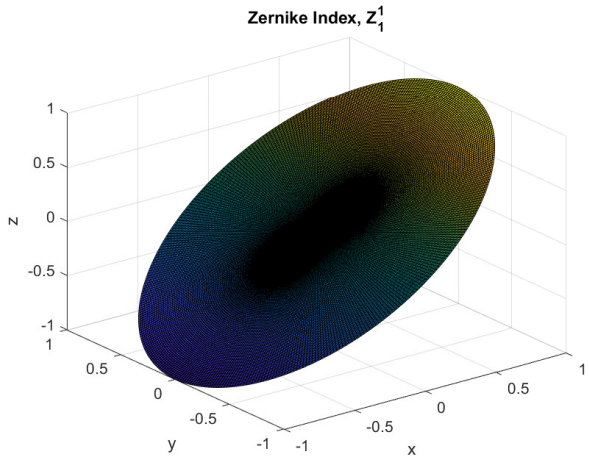

Figure 30: Wavefront for Z_1^1(tip) .

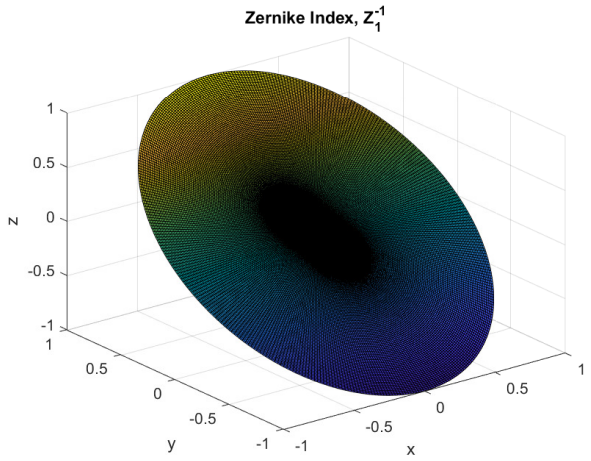

Figure 31: Wavefront for Z_1^{-1}(tilt) .

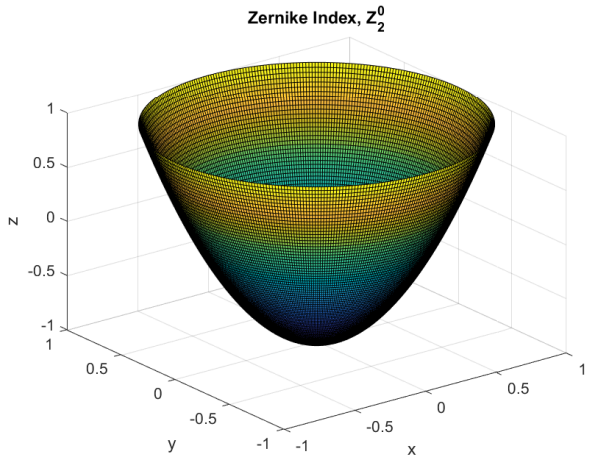

Figure 32: Wavefront for Z_2^0(defocus) .

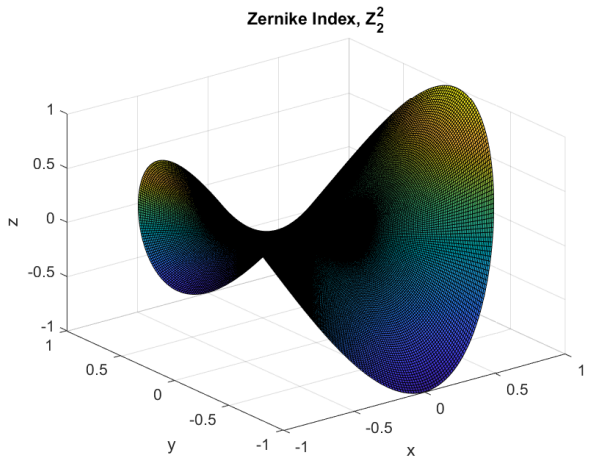

Figure 33: Wavefront for Z_2^2(astigmatism) .

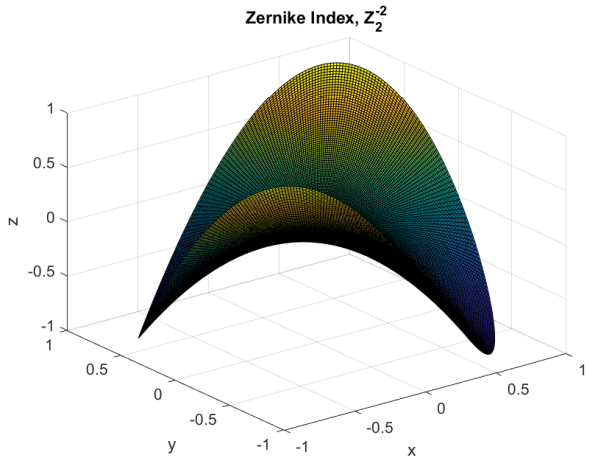

Figure 34: Wavefront for Z_2^{-2}(astigmatism) .

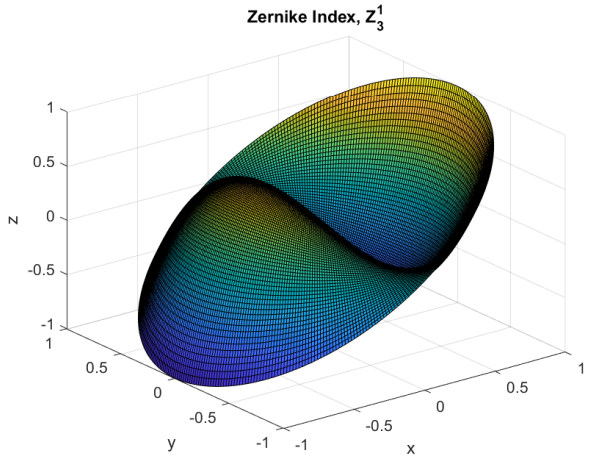

Figure 35: Wavefront for Z_3^1(coma) .

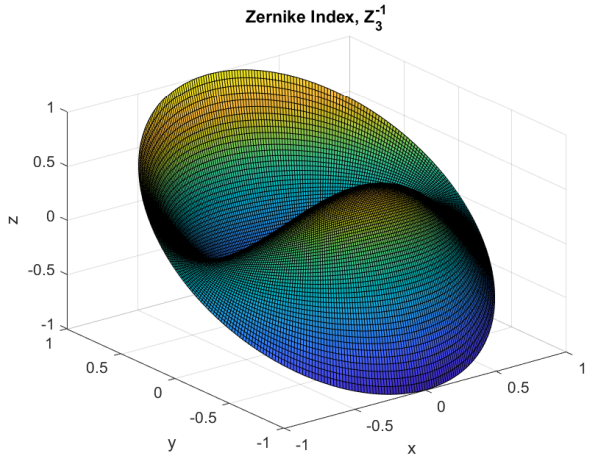

Figure 36: Wavefront for Z_3^{-1}(coma) .

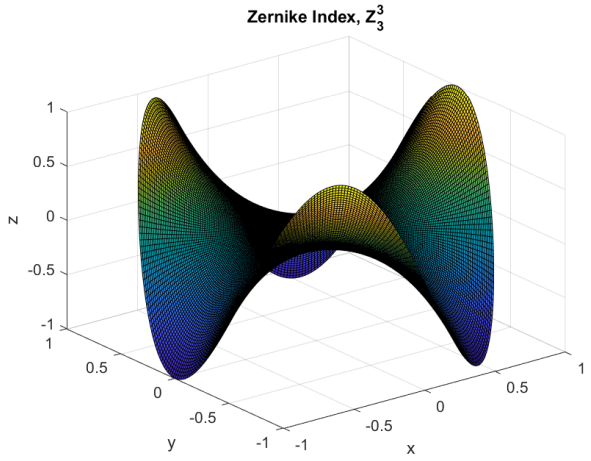

Figure 37: Wavefront for Z_3^3(trefoil) .

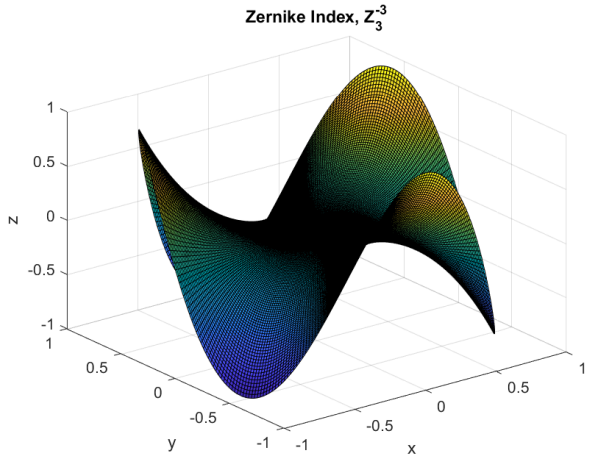

Figure 38: Wavefront for Z_3^{-3}(trefoil) .

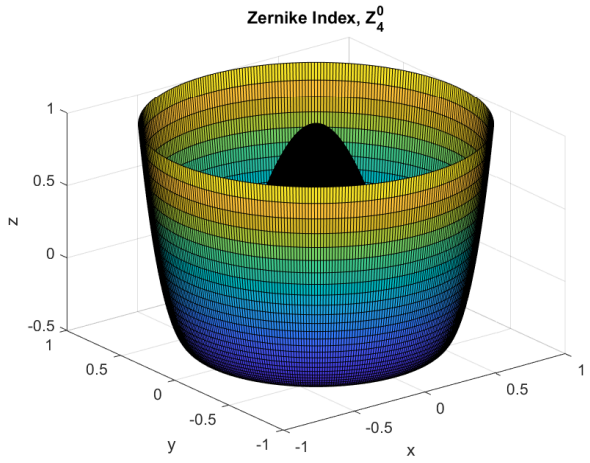

Figure 39: Wavefront for Z_4^0(spherical) .

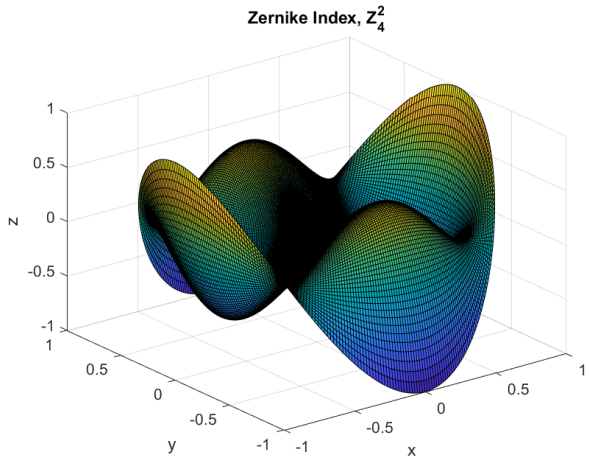

Figure 40: Wavefront for Z_4^2 .

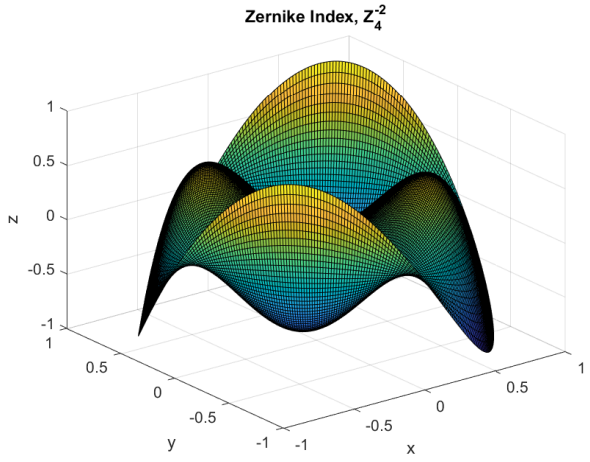

Figure 41: Wavefront for Z_4^{-2} .

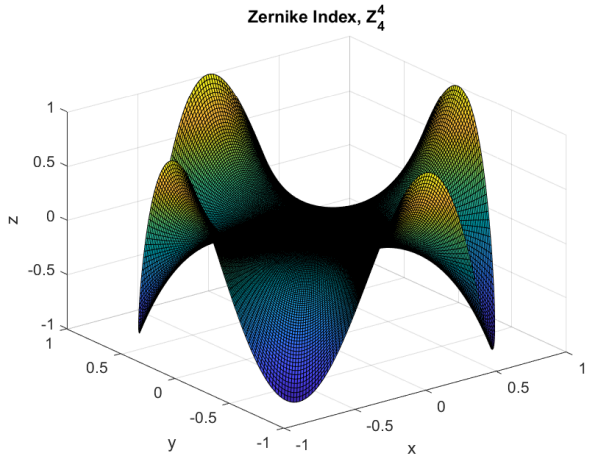

Figure 42: Wavefront for Z_4^4 .

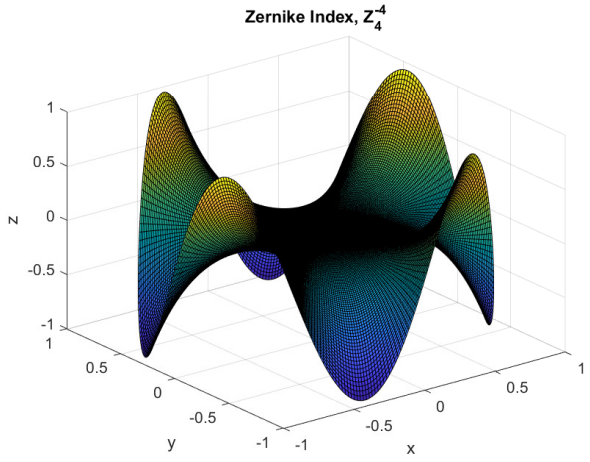

Figure 43: Wavefront for Z_4^{-4}.

4.2 Simulating Zernike in circular apertures

The objective of this section is to construct a function that takes in the size of the circular aperture, geometry of the lenslet array, focal length of the lenslets and simulates the Shack Hartmann pattern for any combination of Zernike co-efficients. An attempt was undertaken to generate the Zernikes with the aid of frequency domain analysis, however due to additional complexity and time limitations the process was discarded. A relatively simpler and hence less accurate method is presented. Some strict assumptions are made, these include focal length of the lenslet at 0.1 unit. CCD sensor with infinite resolution and place exactly at the focal length of the lenses. A minimum space between the lenslets is taken to be 0.03 unit. The aperture mask has a radius of 1 unit. The lenslets have a radius of 1/3 unit. Assumption is made that the displacement of the image point on the image plane is directly proportional to the gradient of the wavefront, as seen in Equation 1. With small angle trigonometric relations for Gaussian lenses this can be expressed as shown in Equation 4.

$$\tan(\alpha) = \frac{\Delta x}{f} = \frac{\delta \phi}{\delta r}$$
Where, α is the angle between nominal and displaced ray.
Thus,
$$\Delta x = f \frac{\delta \phi}{\delta r} \tag{4}$$

With these simplification the results of the simulated Zernikes were presented in Section 3. The main Matlab script along with the functions used are presented in Section 5.

5 Matlab Scripts

Following is the main run script.

```
%% Exercise 4.1, 4.2%%
clc
close all
clear all
%% EX-4.1 : Generate Zernike
rho = [0:0.01:1]; % Define the range of radial vector rho
theta = [0:pi/180:2*pi];% Define the range of theta
[Z_mn, m_feas] = zernike(4, rho, theta); % Return Zernike polynomial values
% over a meshgrid,and the corresponding m values.Each page of Z_mn represents
% each feasible value of m

%% EX - 4.2 : Generate lens shape, aperture + wavefront mask and simulated Zernikes ...
    through the aperture

spacing = 0.03; % minimum space between lenses
lens = [0 0 1/3; (-2/3-spacing) 0 1/3; 0 (2/3 + spacing) 1/3;]; % Three lenslets
lens = [0 0 1/3; (-2/3 - spacing) 0 1/3; (2/3 + spacing) 0 1/3; (1/3+spacing) (2/3 + ...
    spacing) 1/3; (-1/3-spacing) (2/3 + spacing) 1/3 ; (1/3 + spacing) (-2/3 - ...
    spacing) 1/3; (-1/3-spacing) (-2/3-spacing) 1/3 ]; % Hexagonal lenslets


aperture = zeros(size(Z_mn(:,:,1))); % initialising aperture mask
mask = zeros(size(Z_mn(:,:,1))); % initialising mask with Zernike aberration imposed ...
    on the aperture
for k = 1:length(lens(:,1))
    for kk = 1:length(theta)
        for kkk = 1:length(rho)
            [xl, yl] = pol2cart(theta(kk),rho(kkk)); % Cartesian co-ordinate
            if (xl - lens(k,1))^2 + (yl - lens(k,2))^2 <= lens(k,3)^2 % Set lens ...
                aperture active
                aperture(kk,kkk) = 1;
```

```
            end
        mask(kk,kkk) = aperture(kk,kkk)*Z_mn(kk,kkk);  % Generate mask with Zernike ...
            wavefront imposed. Strategy discarded. For visualisation only.
        end
    end
end

%% Select m and N
m = -4;
N = 4;
% Reset rho and theta sample points if required
rho = [0:0.1:1]; % Define the range of radial vector rho
theta = [0:5*pi/180:2*pi];% Define the range of theta
f = 0.05; % focal length
figure()
disp('Lens scatter started')

for k = 1:length(lens(:,1))
    for z = 1:length(rho)
        for zz = 1:length(theta)
            [x,y] = pol2cart(theta(zz), rho(z));
            if (x - lens(k,1))^2 + (y - lens(k,2))^2 < lens(k,3)^2
                [dx,dy] =  zernikegradient(N, m, theta(zz), rho(z)); % Zernike ...
                    gradient estimation in x and y direction
                disturbed = scatter(lens(k,1) - f*dx, lens(k,2) - f*dy, 15, ...
                    'filled','b'); % Plot shifted image points
                hold on
            end
        end
    end
    perfect = scatter(lens(k,1), lens(k,2),100,'r+'); % Plot actual focus point
    lens_bound = viscircles([lens(k,1) lens(k,2)], lens(k,3),'LineStyle','--'); % Plot ...
        lens boundary
    disp(['Simulating lens number: ',num2str(k)])
end
grid on
xlim([-2 2]);
ylim([-2 2]);
disp('lens scatter ends')
legend([disturbed,perfect,lens_bound],'Projected point','Lenslet centroid','Lenslet ...
    boundary','Location','best')
xlabel('x [-]'),ylabel('y [-]')
title(['Effect of Zernike Index Z_{',num2str(N),'}^{', num2str(m),'}(\rho,\theta)']);
hold off
```

Following is the function written to generate the Zernike polynomials.

```
function [zer,m] = zernike(n, rho, theta)
    % Function determinates the
    % Takes in the index n, generates corresponding m automatically.
    % Returns the Zernike matrices Z_{n}^{+-m} (\rho, \theta)
    % # INPUT PARAMETERS: (n, rho, theta)
    % * n, polynomial base index, generate +-m automatically.
    % * rho, radial vector [0,1]
    % * theta, angle vector [0, 2*pi] [rad]
    % # OUTPUT PARAMETERS: [zer,m]
    % * zer = Zernike Polynomial matrix over meshgrid of rho and theta,
    %   rows and columns of meshgrid with values over pages. Each page
    %   defines the m value. first matrix page = first m value.
    % * m = m index
    % * figures autmatically generated
    % # Author: Ali Nawaz
    % * MSc student,"Space Exploration" and "Systems and Control Engineering"
    % * Faculty of Aerospace Engineering,LR, Delft University of Technology.
    % * Faculty of Mechanical, Maritime and Materials Engineering, 3ME Delft
    %   University of Technology.

```

```
    [rho,theta] = meshgrid(rho,theta); % generate meshgrid of rho and theta
    [xbuf, ybuf] = pol2cart(theta, rho); % transform from cylindrical to cartesian ...
        co-ordinates
    syms s; % symbol s for symbolic summation
    m_list = [0:1:n]; % list of m to be selected from
    Z_mn = []; % initialize Zernike [:,:,page] where organised in order of feasible m
    m_feas = []; % select the feasible m

    count = 0; % count the number of instances m is feasible
    for k = 1:length(m_list)
        if mod(n - m_list(k),2)==0
            m_feas = [m_feas,m_list(k)]; % list feasible m
            index1 = (n+m_list(k))/2;
            index2 = (n-m_list(k))/2;
            count = count +1;
            % Find the corresponding radial polynomial
            R_mn(:,:,count) = symsum( ((((-1)^(s))*factorial((n-s)))/( ...
                factorial(s)*factorial(index1 -s)*factorial(index2 - s) ) ...
                ).*rho.^(n-2*s) , s,0, index2  );
        end
    end

% Extending m and R_mn to negative values of m
if mod(n,2) == 0
    m_feas = [-m_feas(2:end),m_feas];
% Zernike polynomial values in form of meshgrid
    Z_mn = cat(3,R_mn(:,:,2:end),R_mn);
elseif mod(n,2) == 1
    m_feas = [-m_feas,m_feas];
% Zernike polynomial values in form of meshgrid
    Z_mn = cat(3,R_mn,R_mn);
end
% Multiplication of azimuthal harmonic component of frequency
    for q = 1:length(m_feas)
        if m_feas(q) > 0
            Z_mn(:,:,q) = Z_mn(:,:,q).*cos(abs(m_feas(q)).*theta);
        elseif m_feas(q) < 0
            Z_mn(:,:,q) = Z_mn(:,:,q).*sin(abs(m_feas(q)).*theta);
        else
            Z_mn(:,:,q) = Z_mn(:,:,q).*1;
        end
    end


    zer = double(Z_mn); % final zernike polynomials in forms of meshgrids
    m = m_feas; % final values of m

    % Plot results
    for r = 1:length(m_feas)
        figure(r)
        surf(xbuf, ybuf,zer(:,:,r))
        title(['Zernike Index, Z_{',num2str(n),'}^{',num2str(m_feas(r)),'}']);
        xlabel('x'),
        ylabel('y')
        zlabel('z')
    end
end
```

Following is the function that returns Zernike gradients upto N= 4.

```
function [dx,dy] = zernikegradient(N, m, theta, rho)
% This return Zernike gradient in cartesian co-ordinate frame
    [x, y] = pol2cart(theta,rho);

    if N == 0
        dx = 0;
        dy = 0;
    elseif N ==1;
```

```
        if m == 1
            dx = 1;
            dy = 0;
        elseif m ==-1
            dx = 0;
            dy = 1;
        elseif m ==0
            dx = 4*x;
            dy = 4*y;
        end
    elseif N ==2
        if m==-2
            dx = 2*y;
            dy = 2*x;
        elseif m ==0
            dx = 4*x;
            dy = 4*y;
        elseif m==2
            dx = 2*x;
            dy = -2*y;
        elseif m== 1
            dx = -2 + 9*x^2 + 3*y^2;
            dy = 6*x*y;
        elseif m ==-1
            dx = 6*x*y;
            dy = -2 + 3*x^2 + 9*y^2;
        end
    elseif N ==3
        if m ==-3
            dx = 6*x*y;
            dy = 3*x^2 - 3*y^2;
        elseif m ==-1
            dx = 6*x*y;
            dy = 9*y^2 + 3*x^2 -2;
        elseif m ==1
            dx = 9*x^2 + 3*y^2 -2;
            dy = 6*x*y;
        elseif m==3
            dx = 3*x^2 - 3 *y^2;
            dy = -6*y*y;
        end
    elseif N == 4
        if m== -4
            dx = 12*(x^2)*y  - 4*y^(3);
            dy = 4*x^3 -12*x*y^2;
        elseif m ==-2
            dx = 24*(x^2)*y  +8*y^(3) - 6*y;
            dy = 8*x^(3) + 24*x*y^2 -6*x;
        elseif m ==0
            dx = 24*x^(3) + 24*x*y^(2) -12*x;
            dy = 24*x^(2) *y + 24*y^(3) - 12*y;
        elseif m==2
            dx = 16*x^3 - 6*x;
            dy = -16*y^4 + 6*y;
        elseif m==4
            dx = 4*x^(3) -12*y^(2)*x;
            dy = 4*y^(3) - 12*x^(2)*y;
        end
    end
end
```

References

[1] MICHEL VERHAEGEN, PAULO POZZI, O. S. G. V. D. W. *Control for High Resolution Imaging*. Lecture notes for the course SC42030 TU Delft, 2017.